Internship proposal

Technical aspects of the creation of a “Coq Platform” and an “OCaml Platform”

Context

OCaml and Coq

OCaml and Coq are two closely-related languages and ecosystems. OCaml is an industrial-strength programming language supporting functional, imperative, and object-oriented styles, with a strong and powerful type-system. Coq is a proof assistant written in OCaml, providing a language to write mathematical statements and their proofs. There are many connections between the two languages and ecosystems: OCaml and Coq originate from the same Inria team where the ancestor of OCaml was created as the implementation language for Coq; even today, the development team and the communities are very much intertwined; Coq supports extraction of verified programs to OCaml; some Coq packages extending core functionalities (called plugins) contain both OCaml and Coq code; several tools are used in both ecosystems / support both languages (the opam package manager, the Dune build system).

Package ecosystems

Code reuse boosts both programmers’ productivity and code quality. This is why successful programming languages now more and more often come with tools to efficiently share and reuse code (package managers) and ecosystems of reusable bricks on which programmers can quickly build (package ecosystems). Both OCaml and Coq have small but thriving package ecosystems, including many renowned libraries such as Base, Cohttp, Lwt, or Yojson in the OCaml case, and Flocq, HoTT, Iris, or Math-Comp in the Coq case. As of today, the only officially supported way of installing these libraries is by using the opam package manager. However, this solution has limitations.

Limitations of opam

One major limitation of opam is that it is not available by default on Windows, although an independently-maintained patched version is already widely used.[1]

Furthermore, opam is not easy to learn for newcomers. Newcomers to OCaml coming from the Javascript / npm world sometimes find it hard to adjust to the opam concepts, which led to the creation of alternatives like esy.[2] But the problem is even more acute for newcomers to Coq which may have even less technical baggage (especially those coming from mathematics).

Finally, opam does not work so well with a system-installed Coq (for instance obtained through Ubuntu’s package repository, or installed using the official installer for macOS), which is how most first-time users install Coq. This leads to a bad overall experience when, not so long after having installed Coq for a first time, users must install Coq a second time (this time using opam) so that they can install packages they need to depend on (such as Math-Comp, a widely used library, even

1 https://github.com/fdopen/opam-repository-mingw
2 https://esy.sh/

for newcomers). The experience is even worse when these users want to use CoqIDE, a standard interface to interact with Coq, which is notoriously hard to install with opam.

Providing a "platform"

A "platform" is a multi-OS distribution of a compiler together with a set of standard packages. Such "platforms" already exist for Haskell[3] and Scala[4] (with an even older example being the TeXLive[5] distribution). A collection of installers is made available to users on any system so that they can all easily get a set of tools and packages installed that will provide a satisfying experience to learn and start using the language. Furthermore, a given version of the platform will provide a unique version of each tool or package on each system. This can be especially useful to teachers having a class of students with varied systems on their laptops, or to researchers who want a reproducible environmment. Ideally, the platform should also include a package manager to allow platform users to start using packages beyond the baseline that it provides.

There have been plans in both OCaml[6] and Coq[7] communities to provide such platforms, and since these two languages and ecosystems are closely related, we wish to explore solutions that would be useful to both at once.

We plan to provide binary installers for Windows, macOS and Linux, as well as an umbrella opam package. Finally, we will encourage package maintainers of Linux distributions to ship the platform as part of their own channels.

Objectives

The intern will explore technical options and produce a prototype for the creation of platforms for the Coq and OCaml ecosystems.

The goal at the end of the internship is to provide a CI / CD (continuous integration / continuous deployment) solution for the continuous creation and distribution of (binary) installers for Windows, macOS, and Linux. The installers for Windows and macOS could reuse the infrastructure already in place to build the Coq installers for Windows and macOS. For Linux, there exist multiple multi-distribution solutions such as Snapcraft,[8] Flatpak,[9] or AppImage.[10] After surveying these options, a choice will be made to target a single such system.

The platform build could rely on the package manager opam or the compositionality features of the Dune build system, both of which are common to OCaml and Coq. An alternative to consider is the esy package manager, which provides a different front-end to the opam registries. It might be important to provide an easy means for platform users to install more packages, or distinct versions, than what the platform provides, and this ability might be related to the underlying choice of technology to build the platform.

The CI solution should allow to continuously test the compatibility of the version of OCaml or Coq in preparation with the latest released (or unreleased) version of each package.

---

3 https://www.haskell.org/platform/
4 https://scalacenter.github.io/platform/platform.html
5 http://www.tug.org/texlive/
6 https://ocaml.org/platform/
7 https://github.com/MSoegtropIMC/coq-platform/blob/master/charter.md
8 https://snapcraft.io/
9 https://flatpak.org/
10 https://appimage.org/

Expected skills, duration

The intern is expected to have previous experience with either OCaml or Coq (ideally both) as a user, and an interest for both. Experience with the opam package manager (and ideally the Dune build system as well) would be a plus. They should also have previous experience with at least one CI service. Experience packaging software for any system is not expected but would be a clear plus.

A good level in English (written) is required.

The expected duration of the internship is 3 to 6 months.

Location, supervision

The internship will take place at IRIF (located in Paris, 13th district) within the PI.R2 Inria project-team and the intern will receive the usual internship stipend of about 500 to 600 euros per month. The main advisors will be Théo Zimmermann (theo@irif.fr) and Yann Régis-Gianas (yrg@irif.fr), in close collaboration with Michael Soegtrop.